VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

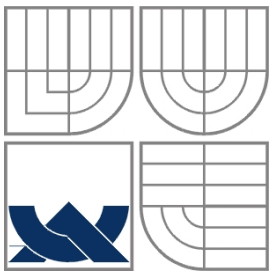
VÝPOČET ALGEBRAICKÝCH ROVNIC

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

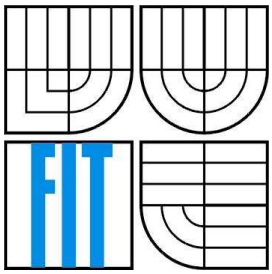
AUTOR PRÁCE
AUTHOR

EVA KUCHAŘOVÁ

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

VÝPOČET ALGEBRAICKÝCH ROVNIC

ALGEBRAIC EQUATIONS CALCULATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

Eva Kuchařová

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Ing. Jiří Kunovský, CSc.

BRNO 2009

Abstrakt

Práce se zabývá řešením soustav lineárních algebraických rovnic metodou převodu na soustavu rovnic diferenciálních, ukáže ale i jiné známější metody. V teoretické části popisuje matematické operace s maticemi. V práci jsou pomocí programů Matlab a TKSL vyřešeny příklady a porovnány jejich hodnoty výsledků. K dispozici jsou zdrojové texty, které se vkládají do programů. Je ukázána funkčnost metody převodu soustavy lineárních rovnic na soustavu rovnic diferenciálních. K vyřešení soustavy diferenciálních rovnic je použita Eulerova metoda. Popisuje se i paralelní řešení soustavy rovnic. V rámci této práce byl vytvořen program lin2dif, který umožňuje převod mezi soustavami rovnic. Jak byl navržen, implementován a jak se s ním pracuje, zahrnuje jedna kapitola této práce.

Klíčová slova

Lineární algebraické rovnice, soustava rovnic, diferenciální rovnice, numerické metody, matice, matematické operace s maticemi, Eulerova metoda, TKSL, Matlab.

Abstract

This work investigates the solution of linear algebraic equations by a conversion to the system of differential equations. Moreover, several more frequently used methods are described in addition. The theoretical part of work describes the mathematical. Next, the examples of using the Matlab a TKSL programs are presented to show and compare the results of calculations. The source codes used are included. The work proves that the solution of algebraic equations by the conversion to the system of differential equations is applicable and further solved by the Euler's method. The parallel solution is also described. As the output of this work, the lin2dif program was developed. It realizes the conversion among the systems of equations. Finally, the description of the work progress, design, implementation and usage of the program are summarized.

Keywords

Linear algebraic equations, system of equations, differential equation, numeric metod, matrix, mathematical operations with matrixes, Euler method, TKSL, Matlab.

Citace

Eva Kuchařová: Výpočet algebraických rovnic. Brno, 2009, bakalářská práce, FIT VUT v Brně.

Výpočet algebraických rovnic

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením Doc. Ing. Jiřího Kunovského, CSc.
Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

……
Eva Kuchařová
20.května 2009

Poděkování

Ráda bych poděkovala Doc. Ing. Jiřímu Kunovskému, CSc. za poskytnuté rady a pomoc při řešení této bakalářské práce.

Obsah

1 Úvod

Tato bakalářská práce byla vypracována jako přehled možností řešení soustav algebraických rovnic. Zaměřuje se také na matematické operace s maticemi, které jsou k vyřešení soustav potřebné. Popisuje různé metody, jak se dopracovat k řešení soustav algebraických rovnic, včetně ukázky práce s programem Matlab. Cílem bylo seznámení se s metodou převodu soustav lineárních algebraických rovnic na soustavu diferenciálních rovnic, její následné řešení, ke kterému je možno využít softwarový prostředek (program TKSL), a vytvořit program, který by převedl soustavu lineárních rovnic na soustavu rovnic diferenciálních bez nutnosti vytváření zdrojových textů.

Práce je rozdělena do sedmi kapitol. Příloha obsahuje manuál k programu lin2dif, který byl vytvořen v rámci této práce, a jeho zdrojový text.

Jelikož se soustavy lineárních rovnic řeší také pomocí matic, je jim věnována druhá kapitola. Obecně se v ní popisují matice a základní matematické operace nad nimi. Pro ilustraci jsou uvedeny příklady.

Třetí kapitola se zabývá metodami řešení soustav lineárních rovnic. Jsou v ní popsané metody přímé, iterační i převod na soustavu rovnic diferenciálních, opět s názornými příklady a také problémy, které mohou vzniknout při řešení.

Ve čtvrté kapitole jsou popsány diferenciální rovnice, jejich řešení pomocí Eulerovy metody, problémy při řešení a paralelní řešení soustav diferenciálních rovnic.

Pátá kapitola přibližuje řešení soustav rovnic pomocí programů Matlab a TKSL. Ukazuje, že metoda řešení soustavy lineárních rovnic pomocí převodu na soustavu diferenciálních rovnic funguje.

Šestá kapitola seznámí s vytvořeným programem lin2dif. Nastíní jeho návrh a práci s ním.

V poslední, sedmé kapitole jsou shrnuty přínosy práce a postřehy při jejím zpracovávání a řešení.

2 Matice

Matice se nejvíce uplatňují při řešení soustav rovnic, kterými se budeme zabývat. Proto si o nich shrneme pár základních informací, které se nám budou při pozdějších výpočtech a převodech mezi soustavami rovnic hodit, a ukážeme si názorné příklady.

Maticí A typu $m \times n$ rozumíme schéma mn reálných nebo komplexních čísel uspořádaných do m vodorovných řad a n svislých sloupců. Obecně můžeme matici zapsat

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ & \ddots & \\ a_{m1} & \dots & a_{mn} \end{pmatrix}$$

Matice mohou být různého typu. Čtvercovou maticí nazýváme takovou matici, když $m = n$, tedy počet řádků se rovná počtu sloupců. Potom hlavní diagonála matice je tvořena čísly $a_{11}, a_{22}, \dots, a_{mn}$. Většinou se ale setkáváme s obdélníkovou maticí, kdy $m \neq n$.

Čtvercovou matici, jejíž determinant je nenulový, označujeme jako matici regulární. Matice, které nejsou regulární, označujeme jako singulární.

Rovnají-li se všechny prvky v matici nule, pak se jedná o nulovou matici. Jednotkovou maticí nazýváme takovou matici, jejíž prvky na hlavní diagonále se rovnají jedné a ostatní prvky jsou nulové.

Dvě matice jsou si rovny, když jsou stejného typu a pro jejich libovolné indexy platí $a_{ij} = b_{ij}$, zjednodušeně řečeno, obě matice obsahují stejné prvky.

Setkat se můžeme i s trojúhelníkovou maticí. Buď s horní trojúhelníkovou maticí, která má všechny prvky pod hlavní diagonálou rovny nule, nebo s dolní trojúhelníkovou matici, jejíž všechny prvky nad hlavní diagonálou jsou nulové.

2.1 Matematické operace

2.1.1 Sčítání matic

Sčítat matice můžeme pouze tehdy, jsou-li stejného typu $m \times n$. Mějme matice A a B, pak jejich součet je matice C, pro kterou platí

$$c_{ij} = a_{ij} + b_{ij} \tag{2.1}$$

kde $i \in \{1,2,\dots,m\}$ a $j \in \{1,2,\dots,n\}$.

Příklad 2.1 Máme-li matice

$$A=\begin{pmatrix}1 & 2 & 5\\ 3 & 7 & 6\\ 5 & -1 & 9\\ 4 & 0 & -2\end{pmatrix}, B=\begin{pmatrix}6 & 1 & 7\\ 8 & 5 & 0\\ -4 & 3 & 2\\ 10 & 7 & 9\end{pmatrix}$$

pak

$$A+B=\begin{pmatrix}1+6 & 2+1 & 5+7\\ 3+8 & 7+5 & 6+0\\ 5+(-4) & -1+3 & 9+2\\ 4+10 & 0+7 & -2+9\end{pmatrix}=\begin{pmatrix}7 & 3 & 12\\ 11 & 12 & 6\\ 1 & 2 & 11\\ 14 & 7 & 7\end{pmatrix}.$$

2.1.2 Násobení matic

Vynásobit matice A a B můžeme pouze tehdy, když je počet sloupců matice A stejný jako počet řádků matice B. Zjednodušeně můžeme říct, že násobíme řádky matice A sloupci matice B. Máme-li tedy matici A typu $m\times p$ a matici B typu $p\times n$, pak jejich vynásobením dostaneme matici C typu $m\times n$, pro kterou platí

$$c_{ij}=a_{i1}b_{1j}+a_{i2}b_{2j}+\ \ \ldots\ \ +a_{ip}b_{pj} \tag{2.2}$$

kde $i\in\{1,2,\ldots,m\}$ a $j\in\{1,2,\ldots,n\}$.

Příklad 2.2 Máme-li matice

$$A=\begin{pmatrix}1 & 2 & 3 & 2\\ 4 & 6 & 5 & -1\end{pmatrix}, B=\begin{pmatrix}8 & 9\\ -2 & 7\\ 4 & -3\\ 5 & 6\end{pmatrix}$$

pak

$$A\cdot B=\begin{pmatrix}1\cdot 8+2\cdot(-2)+3\cdot 4+2\cdot 5 & 1\cdot 9+2\cdot 7+3\cdot(-3)+2\cdot 6\\ 4\cdot 8+6\cdot(-2)+5\cdot 4+(-1)\cdot 5 & 4\cdot 9+6\cdot 7+5\cdot(-3)+(-1)\cdot 6\end{pmatrix}=\begin{pmatrix}26 & 26\\ 35 & 57\end{pmatrix}.$$

2.1.3 Násobení matice číslem

Vynásobíme-li matici A typu $m\times n$ libovolným číslem d, je součinem matice C stejného typu jakým je matice A, tedy $m\times n$. Pro matici C pak platí

$$c_{ij}=d\cdot a_{ij} \tag{2.3}$$

kde $i\in\{1,2,\ldots,m\}$ a $j\in\{1,2,\ldots,n\}$.

Příklad 2.3 Máme-li matici a libovolné číslo

$$A = \begin{pmatrix} 1 & 2 & 5 \\ 3 & 7 & 6 \\ 5 & -1 & 9 \\ 4 & 0 & -2 \end{pmatrix}, d = 5$$

pak

$$A \cdot d = \begin{pmatrix} 5 \cdot 1 & 5 \cdot 2 & 5 \cdot 5 \\ 5 \cdot 3 & 5 \cdot 7 & 5 \cdot 6 \\ 5 \cdot 5 & 5 \cdot (-1) & 5 \cdot 9 \\ 5 \cdot 4 & 5 \cdot 0 & 5 \cdot (-2) \end{pmatrix} = \begin{pmatrix} 5 & 10 & 25 \\ 15 & 35 & 30 \\ 25 & -5 & 45 \\ 20 & 0 & -10 \end{pmatrix}.$$

2.1.4 Odečítání matic

Když umíme sčítat a násobit matice, můžeme tyto znalosti využít k odečtení dvou matic. Rozdílu docílíme tak, že k menšenci přičteme menšitel vynásobený (-1). Stejně jako u součtu matic můžeme odečítat pouze matice stejného typu $m \times n$.

$$A - B = A + (-1) \cdot B \tag{2.4}$$

Příklad 2.4 Máme-li matice

$$A = \begin{pmatrix} 1 & 2 & 5 \\ 3 & 7 & 6 \\ 5 & -1 & 9 \\ 4 & 0 & -2 \end{pmatrix}, B = \begin{pmatrix} 6 & 1 & 7 \\ 8 & 5 & 0 \\ -4 & 3 & 2 \\ 10 & 7 & 9 \end{pmatrix}$$

pak

$$A - B = \begin{pmatrix} 1 + (-1) \cdot 6 & 2 + (-1) \cdot 1 & 5 + (-1) \cdot 7 \\ 3 + (-1) \cdot 8 & 7 + (-1) \cdot 5 & 6 + (-1) \cdot 0 \\ 5 + (-1) \cdot (-4) & -1 + (-1) \cdot 3 & 9 + (-1) \cdot 2 \\ 4 + (-1) \cdot 10 & 0 + (-1) \cdot 7 & -2 + (-1) \cdot 9 \end{pmatrix} = \begin{pmatrix} -5 & 1 & -2 \\ -5 & 2 & 6 \\ 9 & -4 & 7 \\ -6 & -7 & -11 \end{pmatrix}.$$

2.1.5 Transponovaná matice

Transponovaná matice je taková, která vznikne tak, že zaměníme řádky za sloupce. Když máme matici A typu $m \times n$, po transponování se z ní stane matice A^T typu $n \times m$ a pro ni platí

$$a_{ij}^T = a_{ji} \tag{2.5}$$

kde $i \in \{1,2,\ldots,m\}$ a $j \in \{1,2,\ldots,n\}$.

Příklad 2.5 Máme-li matici

$$A=\begin{pmatrix} 4 & -2 & 7 & 8 & -5 \\ 6 & 9 & -3 & 0 & 10 \\ -1 & 5 & 8 & 2 & 7 \end{pmatrix}$$

pak

$$A^T=\begin{pmatrix} 4 & 6 & -1 \\ -2 & 9 & 5 \\ 7 & -3 & 8 \\ 8 & 0 & 2 \\ -5 & 10 & 7 \end{pmatrix}.$$

2.2 Vlastnosti matematických operací

2.2.1 Sčítání

Jak jsme si řekli před chvílí, sčítat matice můžeme pouze ty, které jsou stejného typu. Pro ně pak platí zákon distributivní, komutativní i asociativní. Máme-li matice A, B, C, pak

$$A+(B+C)=(A+B)+C$$
$$A+B=B+A$$
$$A+0=A$$

2.2.2 Násobení

Pro násobení matic platí stejné zákony jako pro sčítání kromě zákonu komutativního.

$$A\cdot 1=A$$
$$(AB)\cdot C=A\cdot(B\cdot C)$$
$$(A+B)\cdot C=A\cdot C+B\cdot C$$
$$A\cdot(B+C)=A\cdot B+A\cdot C$$

Neplatnost komutativního zákonu $A\cdot B\neq B\cdot A$ si budeme demonstrovat na příkladu. Porovnáním výsledků uvidíme, že výsledné matice jsou rozdílné.

Příklad 2.6 Máme-li matice

$$A=\begin{pmatrix} 1 & 2 & 3 & 2 \\ 4 & 6 & 5 & -1 \end{pmatrix}, B=\begin{pmatrix} 8 & 9 \\ -2 & 7 \\ 4 & -3 \\ 5 & 6 \end{pmatrix}$$

pak

$$A \cdot B = \begin{pmatrix} 26 & 26 \\ 35 & 57 \end{pmatrix}, \quad B \cdot A = \begin{pmatrix} 44 & 70 & 69 & 7 \\ 26 & 38 & 29 & -11 \\ -8 & -10 & -3 & 11 \\ 29 & 46 & 45 & 4 \end{pmatrix}.$$

2.2.3 Transpozice

Použít můžeme pouze následující pravidla

$$(A \cdot B)^T = A^T \cdot B^T$$
$$(A + B)^T = A^T + B^T$$

3 Soustava lineárních algebraických rovnic

Soustavu m lineárních algebraických rovnic o n neznámých $x_1, x_2, \ldots, x_n$ můžeme zapsat ve tvaru

$$\begin{array}{llll} a_{11}x_1 + a_{12}x_2 + & \ldots & a_{1n}x_n = b_1 \\ a_{21}x_2 + a_{22}x_2 + & \ldots & a_{2n}x_n = b_2 \\ \vdots \quad \vdots & & \vdots \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + & \ldots & a_{mn}x_n = b_m \end{array}$$

kde a_{mn}, b_m jsou reálná čísla.

K zápisu jednotlivých rovnic můžeme využít indexy i a j. Index i vyjadřuje i-tou rovnici soustavy, v našem případě např.

$$a_{i1}x_1 + a_{i2}x_2 + \quad \ldots \quad + a_{ij}x_j + \quad \ldots \quad + a_{in}x_n = b_j ,$$

a index j označuje j-tou proměnou, např. x_2. Pro indexy platí

$$i \in \{1,2,\ldots,m\} \text{ a } j \in \{1,2,\ldots,n\}.$$

Pak soustavu obecně zapisujeme ve tvaru

$$\sum_{i=1}^{m} a_{ij} \cdot x_j = b_j \tag{3.1}$$

nebo ve tvaru maticovém

$$A \cdot x = B \tag{3.2}$$

kde

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Matici A nazýváme maticí soustavy, x se nazývá vektor neznámých a B vektor pravých stran.

3.1 Řešení soustavy algebraických rovnic

Vyřešit soustavu lineárních algebraických rovnic můžeme pomocí mnoha numerických metod. Ať už to jsou metody přímé (např. Cramerovo pravidlo, Gaussovou eliminační metoda, inverzní matice, …), iterační metody (prostá iterace, Jacobiho metoda, Gauss-Seidlova metoda, …) nebo metoda převodu soustavy lineárních rovnic na soustavu rovnic diferenciálních.

V této práci se budeme zabývat řešením soustav n rovnic o n neznámých

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \quad \ldots \quad a_{1n}x_n &= b_1 \\ a_{21}x_2 + a_{22}x_2 + \quad \ldots \quad a_{2n}x_n &= b_2 \\ \vdots \qquad \vdots \qquad\qquad \vdots \qquad \vdots & \\ a_{n1}x_1 + a_{n2}x_2 + \quad \ldots \quad a_{nn}x_n &= b_n \end{aligned}$$

kde $x_1, \ldots, x_n$ jsou hledané neznámé a dále budeme předpokládat, že matice soustavy je regulární.

3.1.1 Přímé metody

Pomocí přímých metod dostaneme řešení po konečném počtu kroků. Vyvarujeme-li se zaokrouhlovacích chyb, můžeme řešení považovat za přesné.

Cramerovo pravidlo

Máme-li soustavu rovnic, pak každou neznámou x_i můžeme vypočítat pomocí vztahu

$$x_i = \frac{D_i}{D} \tag{3.3}$$

kde D_i je determinant matice soustavy, která vznikne nahrazením i-tého sloupce vektorem pravých stran a D je determinant matice soustavy A.

Abychom mohli použít Cramerovo pravidlo, musí být determinant matice soustavy nenulový a jak uvidíme na následujícím příkladu, toto pravidlo není vhodné používat, máme-li velkou soustavu rovnic, protože s každou rovnicí nám přibývají matematické operace a výpočty jsou složitější.

Příklad 3.1 Vyřešte soustavu rovnic

$$\begin{aligned} 2x_1 - x_2 + x_3 &= 8 \\ x_1 + x_2 + 3x_3 &= 9 \\ x_1 + 2x_2 - x_3 &= 1 \end{aligned}$$

Nejprve vypočítáme všechny determinanty

$$D = \begin{vmatrix} 2 & -1 & 1 \\ 1 & 1 & 3 \\ 1 & 2 & -1 \end{vmatrix} = 2\cdot 1\cdot(-1) + (-1)\cdot 3\cdot 1 + 1\cdot 1\cdot 2 - (1\cdot 1\cdot 1 + 2\cdot 3\cdot 2 + (-1)\cdot 1\cdot(-1)) = -17$$

$$D_1 = \begin{vmatrix} 8 & -1 & 1 \\ 9 & 1 & 3 \\ 1 & 2 & -1 \end{vmatrix} = 8\cdot 1\cdot(-1) + (-1)\cdot 3\cdot 1 + 1\cdot 9\cdot 2 - (1\cdot 1\cdot 1 + 2\cdot 3\cdot 8 + (-1)\cdot 9\cdot(-1)) = -51$$

$$D_2 = \begin{vmatrix} 2 & 8 & 1 \\ 1 & 9 & 3 \\ 1 & 1 & -1 \end{vmatrix} = 2\cdot 9\cdot(-1) + 8\cdot 3\cdot 1 + 1\cdot 1\cdot 1 - (1\cdot 9\cdot 1 + 1\cdot 3\cdot 2 + (-1)\cdot 1\cdot 8) = 0$$

$$D_3 = \begin{vmatrix} 2 & -1 & 8 \\ 1 & 1 & 9 \\ 1 & 2 & 1 \end{vmatrix} = 2\cdot 1\cdot 1 + (-1)\cdot 9\cdot 1 + 8\cdot 1\cdot 2 - (1\cdot 1\cdot 8 + 2\cdot 9\cdot 2 + 1\cdot 1\cdot (-1)) = -34$$

po dosazení do vztahu pro výpočet jednotlivých neznámých pak dostáváme konečné řešení

$$x_1 = \frac{D_1}{D} = \frac{-51}{-17} = 3, \quad x_2 = \frac{D_2}{D} = \frac{0}{-17} = 0, \quad x_3 = \frac{D_3}{D} = \frac{-34}{-17} = 2.$$

Gaussova eliminační metoda

Principem této metody je postupné vylučování jednotlivých neznámých pomocí elementárních úprav z rovnic tak, aby nám v některé rovnici zůstala jenom jedna neznámá, jejíž hodnotu vyčteme. Ostatní neznámé pak získáme dosazením již vypočtených neznámých do příslušných rovnic, tzv. pomocí zpětné substituce.

Při řešení nejdříve získáme rozšířenou matici soustavy. Tj. převedení vektoru pravých stran k matici soustavy a z ní se následně budeme snažit získat matici trojúhelníkového tvaru.

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_{1\,n+1} \\ a_{21} & a_{22} & & a_{2n} & b_{2\,n+1} \\ \vdots & & \ddots & & \vdots \\ a_{n1} & a_{n2} & & a_{nn} & b_{n\,n+1} \end{array}\right)$$

Důležitou podmínkou je, že prvek a_{11} nesmí být nulový. Ale pokud tomu tak je, řádky matice můžeme libovolně přehodit, abychom tuto podmínku splnili, na řešení nám to nic nemění. Dále budeme postupovat, že k druhému až n-tému řádku matice přičítáme násobky prvního řádku matice tak, abychom získali všechny prvky pod prvkem a_{11} nulové. Podobně pokračujeme u dalších prvků, tedy abychom získali pod prvkem a_{22} samé nulové prvky, musíme přičíst násobky druhého řádku k třetímu až n-tému řádku. Takto postupujeme tak dlouho, dokud nedostaneme matici trojúhelníkového tvaru.

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_{1\,n+1} \\ 0 & a_{22} & & a_{2n} & b_{2\,n+1} \\ 0 & 0 & \ddots & & \vdots \\ 0 & 0 & 0 & a_{nn} & b_{n\,n+1} \end{array}\right)$$

Když jsme se dostali k matici trojúhelníkového tvaru, můžeme určovat hodnoty neznámých pomocí následujícího vztahu

$$x_n = \frac{b_{n\,n+1}}{a_{nn}}. \tag{3.4}$$

Než přejdeme k příkladu, měli bychom si říct, že ne vždy je vhodné použít tuto metodu. Jedná se o případ, kdy na řádku, pomocí kterého chceme eliminovat, je nula. Potom bychom museli dělit nulou a to nelze. Proto se provádí tzv. pivotizace, což znamená, že zaměníme n-tý řádek za řádek, který má v n-tém sloupci největší absolutní hodnotu.

Příklad 3.2 Vyřešte soustavu rovnic

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 9 \\ 2x_1 - x_2 + x_3 &= 8 \\ 3x_1 - x_3 &= 3 \end{aligned}$$

Nejdříve vytvoříme rozšířenou matici soustavy

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 9 \\ 2 & -1 & 1 & 8 \\ 3 & 0 & -1 & 3 \end{array}\right)$$

a převedeme ji na matici trojúhelníkového typu

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 9 \\ 2 & -1 & 1 & 8 \\ 3 & 0 & -1 & 3 \end{array}\right) \approx \left(\begin{array}{ccc|c} 1 & 2 & 3 & 9 \\ 0 & 5 & 5 & 10 \\ 0 & 6 & 10 & 24 \end{array}\right) \approx \left(\begin{array}{ccc|c} 1 & 2 & 3 & 9 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & -2 & -6 \end{array}\right).$$

Nyní vidíme, že z třetího řádku můžeme jednoduše vypočítat x_3

$$-2x_3 = -6 \Rightarrow x_3 = 3,$$

z druhého řádku pak pomocí x_3 dopočítáme x_2

$$x_2 + x_3 = 2 \Rightarrow x_2 + 3 = 2 \Rightarrow x_2 = -1$$

a z prvního řádku pomocí x_2 a x_3 dopočítáme x_1

$$x_1 + 2x_2 + 3x_3 = 9 \Rightarrow x_1 + 2 \cdot (-1) + 3 \cdot 3 = 9 \Rightarrow x_1 = 2.$$

Obecně je Gaussova eliminační metoda časově náročná. Když řešíme n rovnic, musíme provést $\frac{n^3}{3}$ aritmetických operací. Proto se hodí spíše pro soustavy, které neobsahují mnoho rovnic.

Gauss-Jordanova metoda

Vychází z Gaussovy eliminační metody. Postup je stejný, jenom nehledáme matici trojúhelníkového tvaru, ale matici diagonálního tvaru. Tzn. prvky hlavní diagonály mají hodnotu 1, ostatní prvky jsou nulové, tedy

$$\left(\begin{array}{ccc|c} 1 & 0 & 0 & b_1 \\ 0 & 1 & 0 & b_2 \\ 0 & 0 & 1 & b_3 \end{array}\right).$$

Můžeme si všimnout, že hodnoty neznámých určíme snáz, ale postup k získání matice diagonálního tvaru je značně pomalejší jako u Gaussovy eliminační metody.

Eliminace s výběrem hlavního prvku

Jedná se opět o pozměněnou Gaussovu eliminační metodu, při jejíž použití by neměly být zaokrouhlovací chyby tak velké.

Při použití metody postupujeme následovně. V prvním kroku vybereme z n-tého sloupce prvek s největší absolutní hodnotou. V dalším kroku řádek, na kterém se prvek s největší absolutní hodnotou nachází, vložíme na místo n-tého řádku matice, pomocí kterého pak eliminujeme. Tento postup opakujeme tak dlouho, dokud nezískáme matici trojúhelníkového typu.

Příklad 3.3 Mějme rozšířenou matici soustavy

$$\left(\begin{array}{ccc|c} 1 & 1 & 2 & -1 \\ 2 & -2 & 1 & -5 \\ \underline{\underline{3}} & 1 & 1 & 3 \end{array}\right)$$

pak k řešení dojdeme následovně. Vybraný hlavní prvek je vždy dvakrát podtržený.

$$\left(\begin{array}{ccc|c} 3 & 1 & 1 & -1 \\ 0 & -2 & -5 & -6 \\ 0 & \underline{\underline{7}} & -2 & 18 \end{array}\right) \approx \left(\begin{array}{ccc|c} 3 & 1 & 1 & -1 \\ 0 & 14 & -4 & -12 \\ 0 & 0 & -39 & 78 \end{array}\right)$$

Zpětnou substitucí získáme výsledek

$$x_1 = 1,\ x_2 = 2,\ x_3 = -2\,.$$

Inverzní matice

Pomocí inverzní matice můžeme řešit pouze soustavy rovnic, které vytvoří čtvercové matice. Máme-li čtvercovou matici A a čtvercovou matici A^{-1}, které jsou ve vztahu $A \cdot A^{-1} = I$, kde I je jednotková matice, pak se matice A^{-1} nazývá inverzní maticí k matici A. Aby měla matice inverzní matici, musí být regulární, tzn. její determinant musí být nenulový. Pak při řešení soustavy lineárních rovnic můžeme použít inverzní matici.

Soustavu rovnic v maticovém tvaru píšeme

$$A \cdot x = B$$

po vynásobení rovnice inverzní maticí A^{-1} a dostáváme

$$A \cdot A^{-1} \cdot x = B \cdot A^{-1}$$

a po úpravě

$$x = b \cdot A^{-1}. \tag{3.5}$$

Pozn. Inverzní matici A^{-1} k matici A můžeme získat pomocí determinantů. Postupujeme následovně:

$$A^{-1} = \begin{pmatrix} \frac{A_{11}}{|A|} & \frac{A_{12}}{|A|} & \cdots & \frac{A_{1n}}{|A|} \\ \frac{A_{2n}}{|A|} & \frac{A_{22}}{|A|} & \cdots & \frac{A_{2n}}{|A|} \\ \vdots & \vdots & \ddots & \\ \frac{A_{n1}}{|A|} & \frac{A_{n2}}{|A|} & & \frac{A_{nn}}{|A|} \end{pmatrix}. \tag{3.6}$$

Jinou možností je rozšířit původní matici A o jednotkovou matici. Vzniklou matici budeme upravovat tak dlouho, dokud se nám na straně matice A neobjeví jednotková matice. Potom jsme došli k řešení a na straně, kde původně byla jednotková matice, dostáváme matici inverzní A^{-1}.

Příklad 3.4 Vyřešte soustavu rovnic

$$\begin{aligned} 2x_1 - x_2 &= 1 \\ -x_1 + 2x_2 &= 2 \end{aligned}$$

Nejprve vytvoříme inverzní matici

$$\left(\begin{array}{cc|cc} 2 & -1 & 1 & 0 \\ -1 & 2 & 0 & 1 \end{array}\right) \leftrightarrow \left(\begin{array}{cc|cc} 2 & -1 & 1 & 0 \\ 0 & \frac{3}{2} & \frac{1}{2} & 1 \end{array}\right) \leftrightarrow \left(\begin{array}{cc|cc} 2 & 0 & \frac{4}{3} & \frac{2}{3} \\ 0 & \frac{3}{2} & \frac{1}{2} & 1 \end{array}\right) \leftrightarrow \left(\begin{array}{cc|cc} 1 & 0 & \frac{2}{3} & \frac{1}{3} \\ 0 & 1 & \frac{1}{3} & \frac{2}{3} \end{array}\right)$$

$$A^{-1} = \begin{pmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{pmatrix}$$

poté dosadíme do vzorce $x = b \cdot A^{-1}$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \cdot \begin{pmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{pmatrix} \Rightarrow \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \frac{4}{3} \\ \frac{5}{3} \end{pmatrix}$$

a dojdeme k řešení

$$x_1 = \frac{4}{3},\ x_2 = \frac{5}{3}.$$

3.1.2 Iterační metody

Při iteračních metodách počítáme přibližné hodnoty řešení. Počet kroků, kterými se dostaneme k řešení pak závisí např. na požadované přesnosti. Těmito metodami je vhodné řešit řídké soustavy rovnic, které mají málo nenulových prvků, protože na rozdíl od přímých metod nebudeme muset provádět tolik matematických operací. Problémem je, že ne vždy dojdeme k výsledku. Důvodem je konvergence metod, která není vždy jednoduchá k určení. Proto bychom měli před použitím iteračních metod zkontrolovat, zda ke konvergenci dojde. Můžeme si také určit maximální počet kroků iterací, které provedeme. Nenacházíme-li řešení, výpočet ukončíme, tedy řekneme, že metoda diverguje.

Obecně je princip iteračních metod jednoduchý. Z první rovnice si vyjádříme první neznámou, z druhé druhou atd., dokud nevyjádříme poslední neznámou. Poté si zvolíme počáteční aproximaci, kterou dosadíme do pravých stran rovnic a vypočítáme další složku aproximace. Analogicky pokračujeme, až splníme podmínky pro ukončení výpočtu.

Vzorec pro prostou iteraci je následující

$$x^{(r+1)} = b - A \cdot x^{r} \tag{3.7}$$

Příklad 3.5 Vyřešte soustavu rovnic

$$5x_1 + x_2 = 7$$
$$x_1 + 5x_2 = 11$$

Nejdříve převedeme soustavu rovnic podle vztahu pro iteraci

$$x_1 = 1{,}4 - 0{,}2x_2$$
$$x_2 = 2{,}2 - 0{,}2x_1$$

pak postupujeme podle popsaného postupu a dostáváme tabulku jednotlivých kroků iterací

kroky iterace	$\mathbf{x_1}$	$\mathbf{x_2}$
1	0	0
2	1,4	2,2
3	0,96	1,92
4	1,016	2,008
5	0,9984	1,9968

Tabulka 3.1 Kroky výpočtu iterací

Pro srovnání máme přesné hodnoty $x_1 = 1, x_2 = 2$. Vidíme, že k řešení se dostáváme již v kroku 5, tedy metoda konverguje. Kdybychom pokračovali v iteracích, výsledek by se zpřesňoval.

Nesmíme zapomenout na podmínku ukončení iterací. V tomto příkladu nám bude stačit výsledek zaokrouhlený na čtyři desetinná místa, nebo určíme maximální počet kroků iterací tj. 5.

Jacobiho metoda

Základní iterační metodou je Jacobiho metoda. Abychom tuto metodu mohli použít, diagonální prvky matice soustavy A nesmí být nulové.

V Jacobiho metodě se používá rozklad matic soustavy A na součet tří matic

$$A = D + L + U \tag{3.8}$$

kde D - diagonální matice, L - dolní trojúhelníková matice, U - horní trojúhelníková matice. Pomocí úprav pak dostáváme vzorec pro Jacobiho iteraci

$$x^{(r+1)} = C \cdot x^r + d \tag{3.9}$$

kde C je tzv. iterační matice Jacobiho metody. Pro prvky c_{ij} matice C a vektor d_i platí

$$c_{ij} = -\frac{a_{ij}}{a_{ii}}, \quad i \neq j, \quad c_{ii} = 0$$

$$d_i = \frac{b_i}{a_{ii}}$$

Vzorec můžeme přepsat ve složkovém zápisu

$$x_i^{(r+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^r \right), \; i \in \{1,2,\ldots,n\} \tag{3.10}$$

Jak již bylo řečeno na začátku podkapitoly, abychom došli k řešení pomocí této metody, musí konvergovat. Podmínkou pro konvergenci je, aby matice soustavy C byla diagonálně dominantní. Tzn. absolutní hodnota v každém řádku matice je větší než součet absolutních hodnot všech ostatních prvků na daném řádku matice

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} |c_{ij}| < |c_{ii}|, \; i \in \{1,2,\ldots,n\} \tag{3.11}$$

a absolutní hodnota v každém sloupci matice je větší než součet absolutních hodnot všech ostatních prvků v daném sloupci matice.

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} |c_{ij}| < |c_{jj}|, \; j \in \{1,2,\ldots,n\} \tag{3.12}$$

Gauss-Seidlova metoda

Postup je stejný jako v Jacobiho metodě. Rozdíl je pouze ten, že k výpočtu další aproximace se použijí nejnovější přibližné hodnoty řešení.

Vzorec pro vyjádření metody ve složkovém zápisu

$$x_i^{(r+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(r+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^r \right),\ i \in \{1,2,\ldots,n\} \tag{3.13}$$

Gauss-Seidlova metoda konverguje, když je matice soustavy diagonálně dominantní nebo pozitivně definitní. Pozitivně definitní matice je taková matice, která je čtvercová a regulární, tj. má nenulový determinant, dokonce platí

$$\det A > 0$$

Můžeme konstatovat, že Gauss-Seidlova metoda konverguje rychleji než metoda Jacobiho. Dále existují takové matice soustav, které konvergují u Gauss-Seidlovy metody, zatímco u Jacobiho metody divergují, totéž platí i naopak. Jacobiho metodu můžeme počítat paralelně, každá neznámá se počítá zvlášť. Gauss-Seidlova metoda je počítána sériově, protože se nejprve musí vypočítat všechny neznámé, aby se mohlo dosazovat, není tak náročná na paměť počítače jako metoda Jacobiho.

Relaxační metody

Slouží k urychlení konvergence Jacobiho a Gauss-Seidlovy metody, kdy si vyjádříme rozdíl mezi dvěmi iteracemi

$$\Delta x_i^r = x_i^{r+1} - x_i^r , \tag{3.14}$$

který spolu s relaxačním parametrem ω změní vzorec pro Gauss-Seidlovu metodu následovně

$$x_i^{(r+1)} = \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(r+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^r \right) + \left(1 - \omega \cdot x_i^r\right) \tag{3.15}$$

kde $i \in \{1,2,\ldots,n\}$ a $\omega \in (0,2)$, obvykle $\omega \in \langle 1,2 \rangle$. Tato metoda se označuje jako superrelaxační.

Budeme-li na pravé straně vzorce psát x_j^r místo $x_j^{(r+1)}$, dostaneme relaxaci Jacobiho metody.

3.1.3 Převod na soustavu diferenciálních rovnic

Tato metoda není tolik používaná a známá jako ostatní, ale i pomocí ní se dají řešit soustavy lineárních rovnic. Můžeme ji zařadit mezi metody iterační, kdy se snažíme nalézt ustálené řešení. K převodu soustavy lineárních algebraických rovnic na soustavu rovnic diferenciálních slouží jednoduchý algoritmus, který budeme demonstrovat na příkladu.

Příklad 3.6 Mějme soustavu lineárních algebraických rovnic

$$x_1 + x_2 + 2x_3 - 5x_4 = 3$$
$$2x_1 + 5x_2 - x_3 - 9x_4 = -3$$
$$2x_1 + x_2 - x_3 + 3x_4 = -11$$
$$x_1 - 3x_2 + 2x_3 + 7x_4 = -5$$

pak v prvním kroku pro převedení soustavy si pro soustavu lineárních rovnic vytvoříme matici soustavy A a matici pravých stran B, tedy

$$A = \begin{pmatrix} 1 & 1 & 2 & -5 \\ 2 & 5 & -1 & -9 \\ 2 & 1 & -1 & 3 \\ 1 & -3 & 2 & 7 \end{pmatrix}, B = \begin{pmatrix} 3 \\ -3 \\ -11 \\ -5 \end{pmatrix}.$$

Následně vytvoříme transponovanou matici A^T k matici soustavy A

$$A^T = \begin{pmatrix} 1 & 2 & 2 & 1 \\ 1 & 5 & 1 & -3 \\ 2 & -1 & -1 & 2 \\ -5 & -9 & 3 & 7 \end{pmatrix}.$$

Předchozí kroky jsme provedli, abychom mohli dosadit do vzorce, díky kterému získáme soustavu diferenciálních rovnic.

$$-x' = A^T \cdot A - A^T \cdot B \tag{3.16}$$

Pro ulehčení počítání vyjádříme původní rovnice pomocí proměnných q_1, q_2, q_3, q_4 a jejich pravé strany převedeme na levou. Tyto proměnné nám vytvoří matici C typu 4×1. Pak nebudeme muset násobit dvě matice typu 4×4 a poté je ještě odečítat od dalšího součinu dvou matic typu 4×4, ale pouze vynásobíme dvě matice typu 4×4 a 4×1, což je jednodušší.

$$q_1 = x_1 + x_2 + 2x_3 - 5x_4 - 3$$
$$q_2 = 2x_1 + 5x_2 - x_3 - 9x_4 + 3$$
$$q_3 = 2x_1 + x_2 - x_3 + 3x_4 + 11$$
$$q_4 = x_1 - 3x_2 + 2x_3 + 7x_4 + 5$$

Použijeme-li matici C, pak se nám vzorec $-x' = A^T \cdot A - A^T \cdot B$ změní na vzorec $-x' = A^T \cdot C$ a dostáváme diferenciální rovnice

$$-x_1' = q_1 + 2q_2 + 2q_3 + q_4$$
$$-x_2' = q_1 + 5q_2 + q_3 - 3q_4$$
$$-x_3' = 2q_1 - q_2 - q_3 + 2q_4$$
$$-x_4' = -5q_1 - 9q_2 + 3q_3 + 7q_4$$

Ať už si počítání pomocí proměnných q_1, q_2, q_3, q_4 ulehčíme nebo ne, v obou případech získáme různými matematickými úpravami soustavu diferenciálních rovnic, kterou už můžeme zadat do výše zmíněných programů (TKSL, Maple, Matlab) k jejímu úplnému vyřešení.

$$\begin{aligned} x_1' &= -10x_1 - 10x_2 + 10x_4 - 30 \\ x_2' &= -10x_1 - 36x_2 + 10x_3 + 68x_4 - 8 \\ x_3' &= 10x_2 - 10x_3 - 10x_4 + 10 \\ x_4' &= 10x_1 + 68x_2 - 10x_3 - 164x_4 - 56 \end{aligned}$$

Nesmíme zapomenout zadat počáteční podmínky, které budou např.

$$x_1(0) = 0, x_2(0) = 0, x_3(0) = 0, x_4(0) = 0$$

Při použití této metody násobíme několik matic. Jsou-li soustavy veliké, může dojít k vysokým nárokům na paměť počítače a čas.

3.2 Problémy při řešení

3.2.1 Zaokrouhlovací chyby

Při řešení lineárních rovnic se zaokrouhlovacích chyb nevyvarujeme téměř nikdy. Jsou způsobené tím, že počítáme se zaokrouhlenými hodnotami, které používáme při dalších výpočtech a jejich výsledky opět zaokrouhlíme, takže se chyby zvyšují.

3.2.2 Podmíněnost soustavy

Numerické úlohy můžeme brát tak, že vstupním hodnotám přiřazujeme výstupní data. Výsledek je hodně citlivý na malé změny ve vstupních hodnotách. Jestliže relativně malým změnám vstupních hodnot odpovídají relativně malé změny výstupních hodnot, pak můžeme říct, že soustava je dobře podmíněná. Zda je soustava dobře podmíněná nám určí číslo podmíněnosti soustavy C_p, které by se mělo co nejvíce blížit k nule.

$$C_p = \frac{\textit{relativní chyba vstupních hodnot}}{\textit{relativní chyba výstupních hodnot}}$$

Na příkladu si ukážeme, jak vypadá špatně podmíněná soustava, kdy relativně malá změna vstupní hodnoty způsobí relativně velkou změnu výstupní hodnoty. Výsledek se nezmění ani volbou jiné metody k vyřešení soustavy.

Příklad 3.7 Mějme soustavu lineárních rovnic

$$2x + 3x = 5$$
$$2x + 3{,}00001y = 5{,}00001$$

jejím vyřešením dostaneme výsledek $x = 1,\ y = 1$.

Zmenšíme-li u druhé rovnice člen y o 0.00002, dostaneme následující soustavu,

$$2x + 3x = 5$$
$$2x + 2{,}99999y = 5{,}00001$$

kterou vyřešíme a dostaneme výsledek $x = 4,\ y = -1$.

3.2.3 Lineárně závislé řádky

Obsahuje-li soustava lineárních rovnic nějakou rovnici, která je násobkem jiné, potom při převodu soustavy na matici soustavy a jejím následným řešením pomocí některé výše popsané metody, dostaneme u těchto rovnic nulové koeficienty a nalezené řešení tak bude nulové.

4 Soustava diferenciálních rovnic

Obyčejná diferenciální rovnice n-tého řádu se zapisuje

$$y^{(n)} = f\left(x, y, y', \ldots, y^{(n=1)}\right) \tag{4.1}$$

nesmíme zapomenout na počáteční podmínky $y(x_0) = y_0, y'(x_0) = y_0', \ldots, y^{(n=1)}(x_0) = y^{(n=1)}$.

Diferenciální rovnici n-tého řádu můžeme převést na rovnici prvního řádu pomocí substitucí

$$z = y, z_1 = y', \ldots, z_n = y^{n-1} \tag{4.2}$$

výsledná soustava n rovnic prvního řádu pak bude

$$\begin{aligned} y_1' &= z_1, \quad y_1(x_0) = y_0 \\ y_2' &= z_2, \quad y_2(x_0) = y_1 \\ &\vdots \\ y_n' &= f(x, z, z_1, z_2, \ldots, z_n) \quad y_n(x_0) = y_0^{n=1} \end{aligned} \tag{4.3}$$

4.1 Řešení diferenciálních rovnic

Při řešení diferenciálních rovnic a jejich soustav nás většinou nezajímá spojitá funkce jako výsledek, ale pouze přibližné hodnoty řešení. Ty získáme pomocí bodů $x_0, \ldots, x_n$, které si vygenerujeme, pomocí numerických metod vypočítáme body $y_0, y_1, \ldots y_n$. Výsledné přibližné hodnoty řešení pak můžeme porovnat s hodnotami přesného řešení tj. $y(x_0), y(x_1), y(x_2), \ldots, y(x_n)$.

V této práci si popíšeme Eulerovu metodu řešení diferenciálních rovnic a následně paralelní řešení soustavy diferenciálních rovnic.

4.1.1 Eulerova metoda

Jedná se o nejjednodušší ale ne příliš přesnou metodu prvního řádu k řešení diferenciálních rovnic. Vyjádřit ji můžeme vzorcem

$$y_{(i+1)} = y_i + h \cdot f(x_i, y_i), \tag{4.4}$$

kde h vyjadřuje délku kroku metody.

Pokud zohledníme i lokální chybu $L = \frac{h^2}{2} \cdot y'' + \ldots$ a $f(x_i, y_i)$ přepíšeme na $y'(x_i)$ získáme vzorec

$$y_{(i+1)} = y(x_i) + h \cdot y'(x_i) + \frac{h^2}{2} \cdot y'' + \ldots \tag{4.5}$$

Důvodem ne příliš velké přesnosti je, že při počítání následujícího kroku z x_i do x_{i+1} používáme na celém intervalu konstantní hodnotu derivace $f(x_i, y_i)$, která se však v průběhu kroku mění. Abychom dosáhli pomocí Eulerovy metody přesnějších výsledků, měli bychom počítat s co nejmenším krokem.

Příklad 4.1 Vypočtěte pomocí Eulerovy metody rovnici na intervalu $x \in \langle 0,\, 0{,}6 \rangle$

$$y' = x^2 - y\,,\ y(0) = 1.$$

Abychom ukázali, že je lepší zvolit menší krok, vyřešíme metodu s kroky $h_1 = 0{,}1$, $h_2 = 0{,}2$, $h_3 = 0{,}3$. K dispozici budeme mít také přesné hodnoty řešení k porovnání s přibližnými hodnotami.

K řešení použijeme vzorec $y_{(i+1)} = y_i + h \cdot f(x_i, y_i)$. Po dosazení dostáváme následující hodnoty neznámých:

řešení s krokem h_1 :

$$\begin{aligned}
y_1 &= y_0 + h_1 \cdot (x_0^2 - y_0) = 1 + 0{,}1 \cdot (0^2 - 1) = 0{,}9 \\
y_2 &= y_1 + h_1 \cdot (x_1^2 - y_1) = 0{,}9 + 0{,}1 \cdot (0{,}1^2 - 0{,}9) = 0{,}811 \\
y_3 &= y_2 + h_1 \cdot (x_2^2 - y_2) = 0{,}811 + 0{,}1 \cdot (0{,}2^2 - 0{,}811) = 0{,}7339 \\
y_4 &= y_3 + h_1 \cdot (x_3^2 - y_3) = 0{,}7339 + 0{,}1 \cdot (0{,}3^2 - 0{,}7339) = 0{,}6695 \\
y_5 &= y_4 + h_1 \cdot (x_4^2 - y_4) = 0{,}6695 + 0{,}1 \cdot (0{,}4^2 - 0{,}6695) = 0{,}6186 \\
y_6 &= y_5 + h_1 \cdot (x_5^2 - y_5) = 0{,}6186 + 0{,}1 \cdot (0{,}5^2 - 0{,}6186) = 0{,}5817
\end{aligned}$$

řešení s krokem h_2:

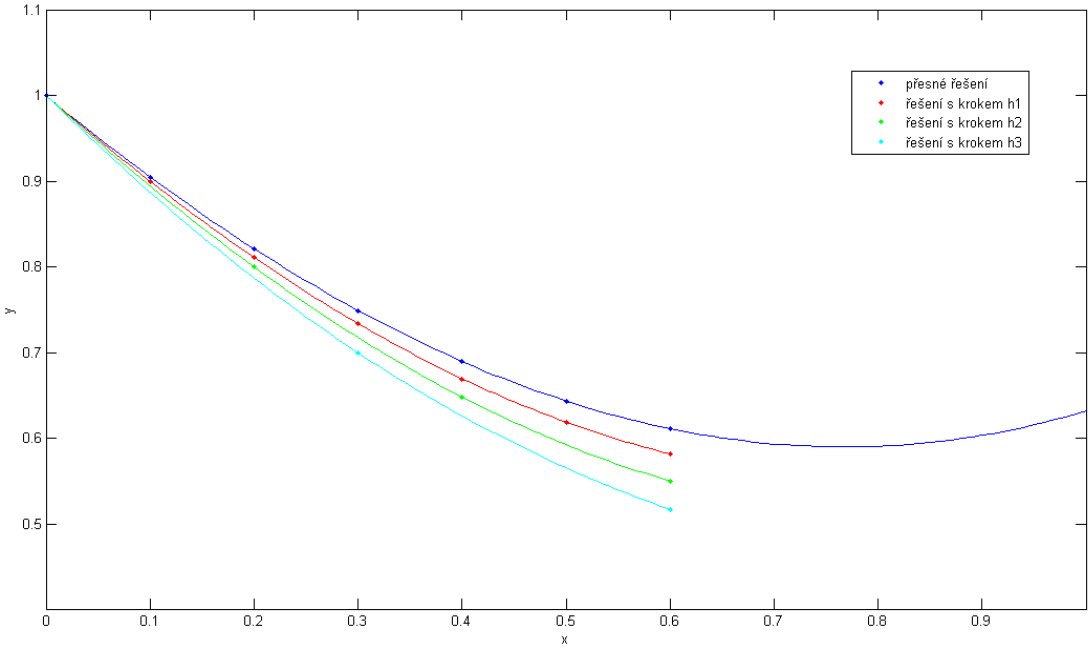
$$\begin{aligned}
y_1 &= y_0 + h_2 \cdot (x_0^2 - y_0) = 1 + 0{,}2 \cdot (0^2 - 1) = 0{,}8 \\
y_2 &= y_1 + h_2 \cdot (x_1^2 - y_1) = 0.8 + 0{,}2 \cdot (0{,}2^2 - 0{,}8) = 0{,}648 \\
y_3 &= y_2 + h_2 \cdot (x_2^2 - y_2) = 0{,}648 + 0{,}2 \cdot (0{,}4^2 - 0{,}648) = 0{,}5504
\end{aligned}$$

řešení s krokem h_3:

$$\begin{aligned}
y_1 &= y_0 + h_3 \cdot (x_0^2 - y_0) = 1 + 0{,}3 \cdot (0^2 - 1) = 0{,}7 \\
y_2 &= y_1 + h_3 \cdot (x_1^2 - y_1) = 0.7 + 0{,}3 \cdot (0{,}3^2 - 0{,}7) = 0{,}517
\end{aligned}$$

přesné hodnoty:

$$y_1 = 0{,}9052$$
$$y_2 = 0{,}8213$$
$$y_3 = 0{,}7492$$
$$y_4 = 0{,}6897$$
$$y_5 = 0{,}6435$$
$$y_6 = 0{,}6112$$



Obrázek 4.1 Výsledný graf řešení pro porovnání jednotlivých výsledků

4.2 Paralelní řešení soustavy diferenciálních rovnic

Soustavu vyřešíme pomocí Eulerovy metody. V jednotlivých krocích metody můžeme rovnice počítat nezávisle na sobě, tj. paralelně. Řešení ukážeme na příkladu.

Příklad 4.2 Mějme soustavu diferenciálních rovnic s počátečními podmínkami

$$x' = -x - y + 1$$
$$y' = x - y - 3$$
$$x(0) = 0,\ y(0) = 0$$

pak postupujeme následovně. Z rovnic vypočítáme x_1 a y_1. Jejich hodnoty pak dosadíme do dalšího kroku metody, jeho výsledky pak dosadíme opět do dalšího kroku, postup opakujeme, dokud se nám řešení neustálí. Krok Eulerovy metody si zvolíme libovolně, např. $h = 0{,}1$.

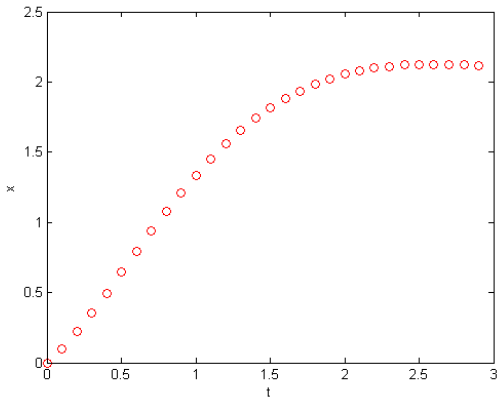
$$x_1 = x_0 + h \cdot (-x_0 - y_0 + 1) = 0 + 0{,}1 \cdot (-0 - 0 + 1) = 0{,}1$$
$$y_1 = y_0 + h \cdot (x_0 - y_0 - 3) = 0 + 0{,}1 \cdot (0 - 0 - 3) = -0{,}3$$
$$x_2 = x_1 + 0{,}1 \cdot (-x_1 - y_1 + 1) = 0{,}1 + 0{,}1 \cdot (-0{,}1 + 0{,}3 + 1) = 0{,}16$$
$$y_2 = y_1 + h \cdot (x_1 - y_1 - 3) = -0{,}3 + 0{,}1 \cdot (0{,}1 + 0{,}3 - 3) = -0{,}56$$
$$\vdots$$

atd.

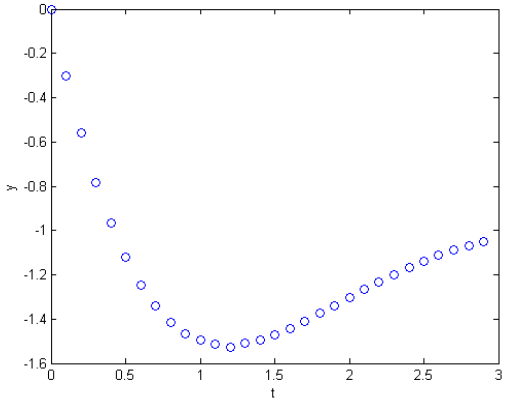
Z prvním dvou kroků je zřejmé, jak se postupuje, proto nebudeme vypisovat všechny výpočty do ustáleného řešení, ale postačí nám tabulka s vyřešenými hodnotami. Tabulka neobsahuje hodnoty až do ustáleného řešení, ale pro ilustraci to stačí. Vidíme, že k ustálenému řešení už moc kroků nezbývá.

$\mathbf{x_0}$	0	$\mathbf{y_0}$	0	$\mathbf{x_{15}}$	1,8153	$\mathbf{y_{15}}$	-1,4698
$\mathbf{x_1}$	0,1000	$\mathbf{y_1}$	-0,3000	$\mathbf{x_{16}}$	1,8808	$\mathbf{y_{16}}$	-1,4413
$\mathbf{x_2}$	0,2200	$\mathbf{y_2}$	-0,5600	$\mathbf{x_{17}}$	1,9368	$\mathbf{y_{17}}$	-1,4091
$\mathbf{x_3}$	0,3540	$\mathbf{y_3}$	-0,7820	$\mathbf{x_{18}}$	1,9840	$\mathbf{y_{18}}$	-1,3745
$\mathbf{x_4}$	0,4968	$\mathbf{y_4}$	-0,9684	$\mathbf{x_{19}}$	2,0231	$\mathbf{y_{19}}$	-1,3386
$\mathbf{x_5}$	0,6440	$\mathbf{y_5}$	-1,1219	$\mathbf{x_{20}}$	2,0546	$\mathbf{y_{20}}$	-1,3024
$\mathbf{x_6}$	0,7918	$\mathbf{y_6}$	-1,2453	$\mathbf{x_{21}}$	2,0794	$\mathbf{y_{21}}$	-1,2667
$\mathbf{x_7}$	0,9371	$\mathbf{y_7}$	-1,3416	$\mathbf{x_{22}}$	2,981	$\mathbf{y_{22}}$	-1,2321
$\mathbf{x_8}$	1,0776	$\mathbf{y_8}$	-1,4137	$\mathbf{x_{23}}$	2,1115	$\mathbf{y_{23}}$	-1,1991
$\mathbf{x_9}$	1,2112	$\mathbf{y_9}$	-1,4646	$\mathbf{x_{24}}$	2,1203	$\mathbf{y_{24}}$	-1,1680
$\mathbf{x_{10}}$	1,3365	$\mathbf{y_{10}}$	-1,4970	$\mathbf{x_{25}}$	2,1251	$\mathbf{y_{25}}$	-1,1392
$\mathbf{x_{11}}$	1,4526	$\mathbf{y_{11}}$	-1,5137	$\mathbf{x_{26}}$	2,1265	$\mathbf{y_{26}}$	-1,1128
$\mathbf{x_{12}}$	1,5587	$\mathbf{y_{12}}$	-1,5170	$\mathbf{x_{27}}$	2,1251	$\mathbf{y_{27}}$	-1,0888
$\mathbf{x_{13}}$	1,6545	$\mathbf{y_{13}}$	-1,5095	$\mathbf{x_{28}}$	2,1215	$\mathbf{y_{28}}$	-1,0675
$\mathbf{x_{14}}$	1,7400	$\mathbf{y_{14}}$	-1,4931	$\mathbf{x_{29}}$	2,1161	$\mathbf{y_{29}}$	-1,0486

Tabulka 4.1 Kroky výpočtu Eulerovou metodou

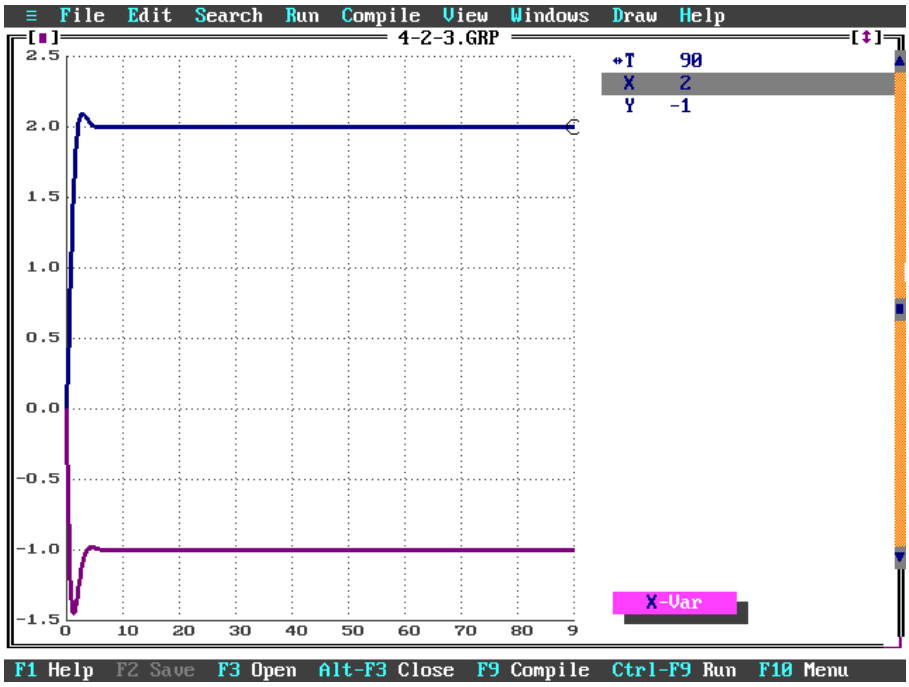


Obrázek 4.2 Grafické řešení x



Obrázek 4.3 Grafické řešení y

Pro srovnání jsme soustavu rovnic vyřešili i v programu TKSL, který se k výsledku dopracuje použitím metody Taylorova typu.



Obrázek 4.4 Grafické řešení programem TKSL

4.3 Problémy při řešení

4.3.1 Lokální a globální chyba

Lokální chyby se můžeme dopustit při jednokrokových metodách, předpokládáme-li, že všechny hodnoty, které jsme při počítání použili, byly přesné. Vyjádřit ji můžeme jako rozdíl přibližného řešení v daném bodě x_i a řešení, které splňuje podmínku $y(x_{i-1}) = y_{i-1}$.

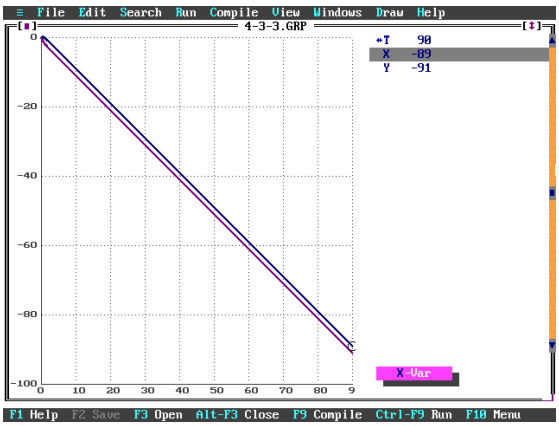
V této práci byla popsána pouze Eulerova metoda, tj. jednokroková metoda. Ale existují také vícekrokové metody při kterých se v dalším kroku nepoužívají hodnoty pouze z jednoho předchozího kroku, ale z více kroků. U těchto vícekrokových metod se vyskytuje globální chyba, která vznikne nahromaděním se více lokálních chyb.

4.3.2 Nestabilita soustavy

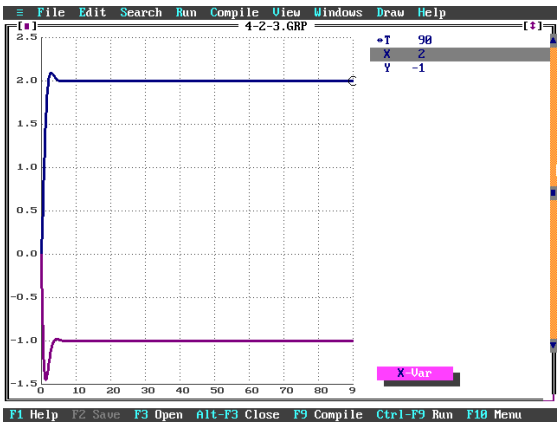
Není-li matice soustavy pozitivně definitní, vyjde nám nestabilní řešení, tj. nepřesné. Můžeme konstatovat, že nestabilní soustava je taková, jejíž determinant matice soustavy se blíží nule.

Stabilní řešení je takové, které se ustálí na určité hodnotě, tedy používáme-li krokové metody, pak výsledky následujících kroků jsou po určité době konstantní.

Máme-li soustavu diferenciálních rovnic, kterou jsme získali převodem ze soustavy lineárních rovnic pomocí algoritmu popsaného v kapitole 3.1.3, dostaneme vždy stabilní soustavu. Důvodem je násobení transponovanou maticí.



Obrázek 4.5 Nestabilní řešení



Obrázek 4.6 Stabilní řešení

4.3.3 Konvergence metody

Stejně jako u řešení lineárních rovnic, tak i u rovnic diferenciálních požadujeme, aby přibližné řešení konvergovalo. Přesnost metody, nám ovlivňuje použitá délka kroku h, viz. (4.1.1 Eulerova metoda). Aby řešení konvergovalo k přesnému řešení, měl by se použitý krok h blížit co nejvíce k nule.

Řádem metody je označována rychlost konvergence metody. Je to takové přirozené číslo p, pro které platí, že je-li h malé, pak velikost lokální chyby d_i je h^{p+1}.

5 Programy umožňující výpočet soustav rovnic

5.1 Program TKSL

Program TKSL umožňuje počítat soustavy diferenciálních rovnic. K řešení používá metodu Taylorova typu. Práce s programem je jednoduchá.

Příklad 5.1 Mějme soustavu lineárních rovnic

$$2x_1 - x_2 = 3$$
$$3x_1 + 4x_2 = 10$$

Pomocí programu lin2dif ji převedeme na soustavu rovnic diferenciálních

$$x_1' = -13x_1 - 10x_2 + 36$$
$$x_2' = -10x_1 - 17x_2 + 37$$

určíme počáteční podmínky, krok metody, dobu výpočtu a vyřešíme programem TKSL. Soustavu diferenciálních rovnic můžeme do programu napsat ručně nebo ji načíst ze souboru. Zápis zdrojového kódu je následující

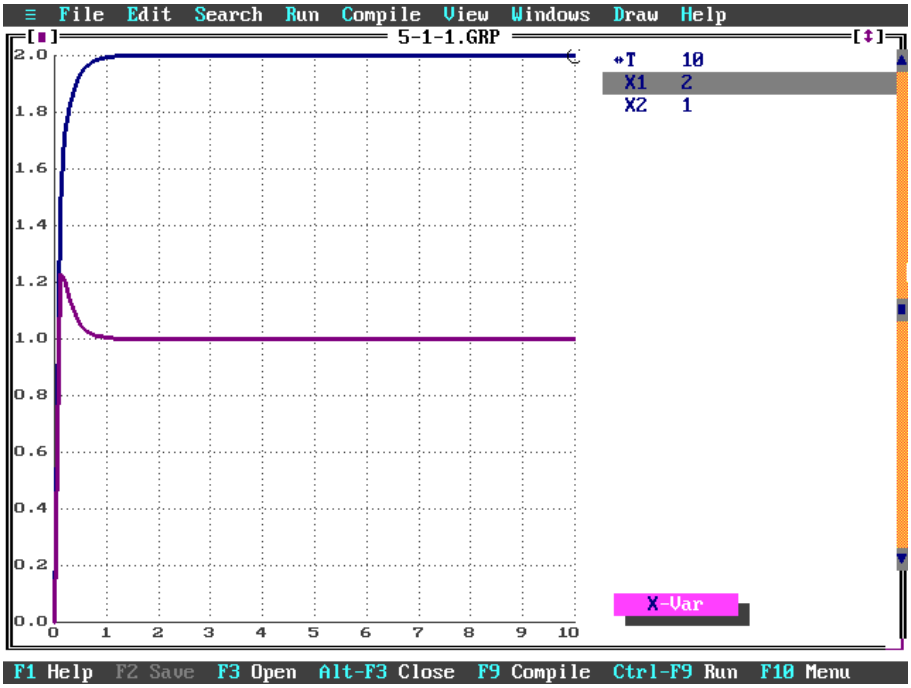
```
var x1,x2;
      const dt=0.1,tmax=10,eps=1e-20;

            system

                  x1' =  -13*x1 -10*x2 + 36 &0;
                  x2' =  -10*x1 -17*x2 + 37 &0;

            sysend.
```

Poté stačí zadat příkaz run v horním menu a vykreslí se výsledný graf s hodnotami řešení.



Obrázek 5.1 Grafické řešení programem TSKL

Na dalším příkladu ukážeme, že ne vždy nám vyjdou přesné hodnoty. Zatímco v podkapitole 5.2, kde tu samou soustavu rovnic řešíme v programu Matlab, ano.

Příklad 5.2 Mějme soustavu lineárních rovnic:

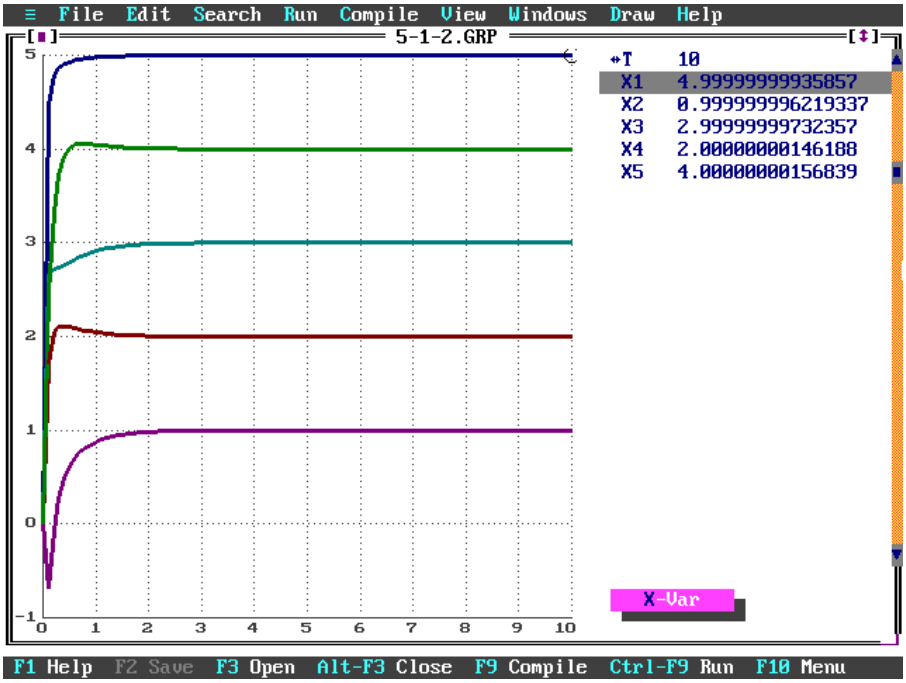
$$\begin{aligned} 2x_1 + x_2 - x_3 + 4x_4 - x_5 &= 12 \\ x_1 - 2x_2 + x_3 - x_4 + 2x_5 &= 12 \\ 3x_1 + x_2 - 2x_3 - 2x_4 + x_5 &= 10 \\ 4x_1 - 2x_2 + x_3 - x_4 - x_5 &= 15 \\ x_1 - x_2 + 4x_3 + x_4 + 2x_5 &= 26 \end{aligned}$$

Postup bude stejný jako v předchozím příkladu. Pro názornost opět uvedeme zdrojový kód programu TKSL

```
var x1,x2,x3,x4,x5;
      const dt=0.1,tmax=10,eps=1e-20;

      system
                  x1' =  -31*x1 + 6*x2 -1*x3 + 2*x4 -1*x5 + 152 &0;
                  x2' =  6*x1 -11*x2 + 11*x3 -5*x4 + 4*x5 -58 &0;
                  x3' =  -1*x1 + 11*x2 -23*x3 -2*x4 -8*x5 + 99 &0;
                  x4' =  2*x1 -5*x2 -2*x3 -23*x4 + 5*x5 + 27 &0;
                  x5' =  -1*x1 + 4*x2 -8*x3 + 5*x4 -11*x5 + 59 &0;

            sysend.
```



Obrázek 5.2 Grafické řešení programem TKSL

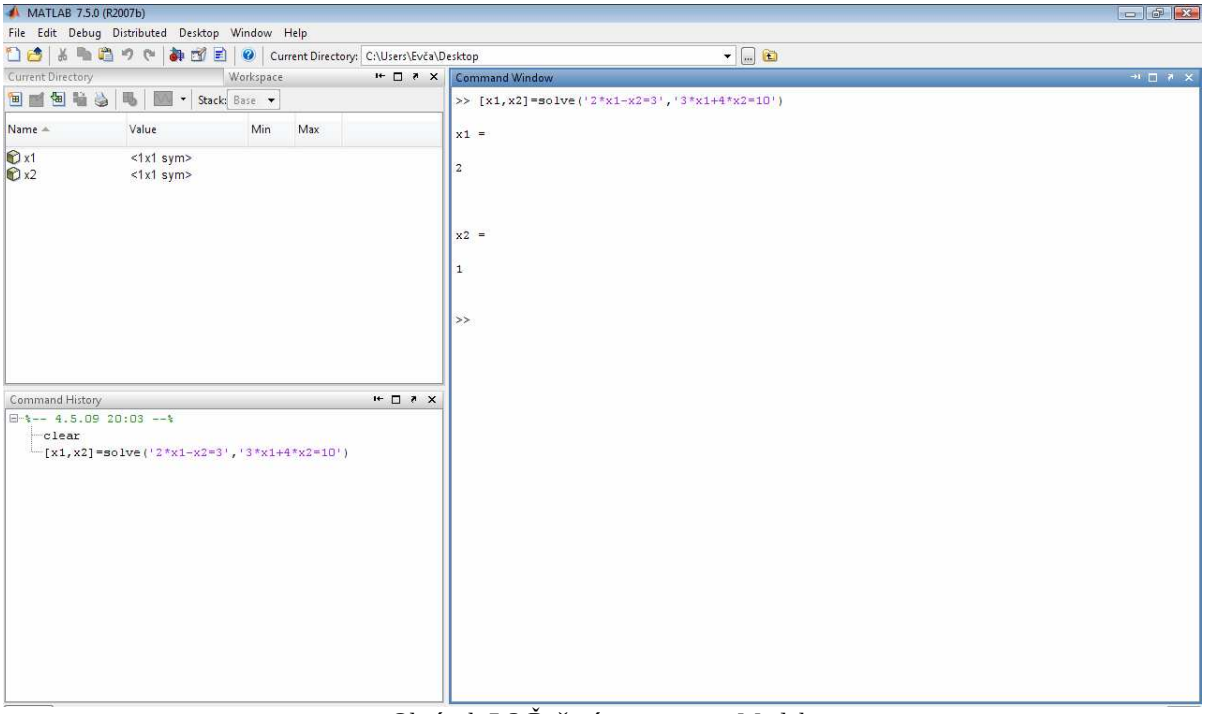
5.2 Program Matlab

V programu Matlab vyřešíme stejné soustavy lineárních rovnic jako v kapitole 5.1. Důvodem je, abychom dokázali, že řešíme-li soustavu rovnic lineárních rovnic převodem na soustavu rovnic diferenciálních, dostaneme stejné, resp. přibližné výsledky, jako když řešíme soustavu lineárních rovnic numericky nebo analyticky.

K řešení lineárních rovnic se v programu Matlab používá funkce *solve*. Nejprve se program snaží vyřešit soustavu rovnic analyticky a pokud nenalezne řešení, přejde k metodám numerickým.

Vyřešením příkladu 5.1 nalezneme hodnoty řešení $x_1 = 2, x_2 = 1$. Do programu se napíše následující kód

```
[x1,x2]= solve('2*x1-x2=3',
    '3*x1+4*x2=10')
```



Obrázek 5.3 Řešení v programu Matlab

Vyřešíme i příklad 5.2, uvedeme zápis kódu do programu Matlab a spočítané hodnoty řešení, které jsou $x_1 = 5, x_2 = 1, x_3 = 3, x_4 = 2, x_5 = 4$.

```
[x1,x2,x3,x4,x5]= solve('2*x1+x2-x3+4*x4-x5=12',
'x1-2*x2+x3-x4+2*x5=12',
'3*x1+x2-2*x3-2*x4+x5=10',
'4*x1-2*x2+x3-x4-x5=15',
'x1-x2+4*x3+x4+2*x5=26')
```

5.3 Srovnání TKSL a Matlabu

Některé soustavy lineárních rovnic, které jsme zadali v programu Matlab, byly vyřešeny přesněji než v programu TKSL za pomoci převodu na soustavu diferenciálních rovnic. Při řešení špatně podmíněných soustav rovnic jsme v programu TKSL neměli problém získat výsledky, zatímco v programu Matlab jsme se k výsledkům nedopracovali.

Závěrem můžeme říct, že pravděpodobnost získání výsledků při řešení soustav lineárních rovnic je větší, řešíme-li je pomocí programu TKSL s využitím algoritmu převodu na soustavu diferenciálních rovnic.

6 Program lin2dif

Jedním z cílů této bakalářské práce bylo vytvořit jednoduchý program, který by převedl zadanou soustavu lineárních rovnic na soustavu rovnic diferenciálních. Program pak ulehčí mnohdy zdlouhavé počítání při převodu mezi jednotlivými soustavami.

6.1 Požadavky

Hlavním požadavkem bylo spuštění aplikace v operačním systému Windows. Dále, aby při velkém počtu rovnic, uživatel nemusel zadávat všechny prvky, tzn. nulové prvky matice se zadávat nemusí. Jelikož program bude sloužit spíše demonstračně, pracuje pouze s celými čísly. Maximální počet lineárních rovnic, který může být zadaný, byl stanoven na 100. Posledním požadavkem bylo uložení převedené soustavy do textového souboru.

6.2 Implementace

Při navrhování se předpokládalo, že uživatel rozumí dané problematice, proto se zadávají pouze prvky matic, ne celé lineární rovnice.

Program byl napsán v jazyce C++ v programovacím prostředí Microsoft Visual Studio. V celém programu se pracuje s maticemi a provádí se s nimi takové matematické operace, aby se získal požadovaný výsledek.

Nejdříve jsou v programu deklarované dvě statické matice obsahující nulové prvky. Matice A o rozměrech 100x100, která představuje matici soustavy, a matice B o rozměrech 100x1 představující vektor pravých stran. Dále tři funkce, které nám provedou operace s maticemi - *transponuj_matici*, *nasobeni_matice*, *neguj_matici*. Načítání jednotlivých prvků matice je řešeno pomocí standardního vstupu ve funkci *nacti_matici*.

Jakmile jsou matice načteny, zkopíruje se matice A do matice T pomocí funkce *kopiruj_matici*. Matice T je následně transponována. Poté jsou vytvořeny další dvě matice C (násobek T a A) a D (násobek T a B), jejichž prvky jsou použity do výpisu výsledku. Pro zobrazení výsledku slouží funkce *vypis_vysledek*, která volá funkci *setRadky*. Ta za sebe vypíše nenulové prvky matice C, ke kterým přidá příslušnou neznámou, a prvky matice D. Pro výpis do souboru vysledek.txt slouží funkce *vypis_vysledek_do* využívající také funkci *setRadky*.

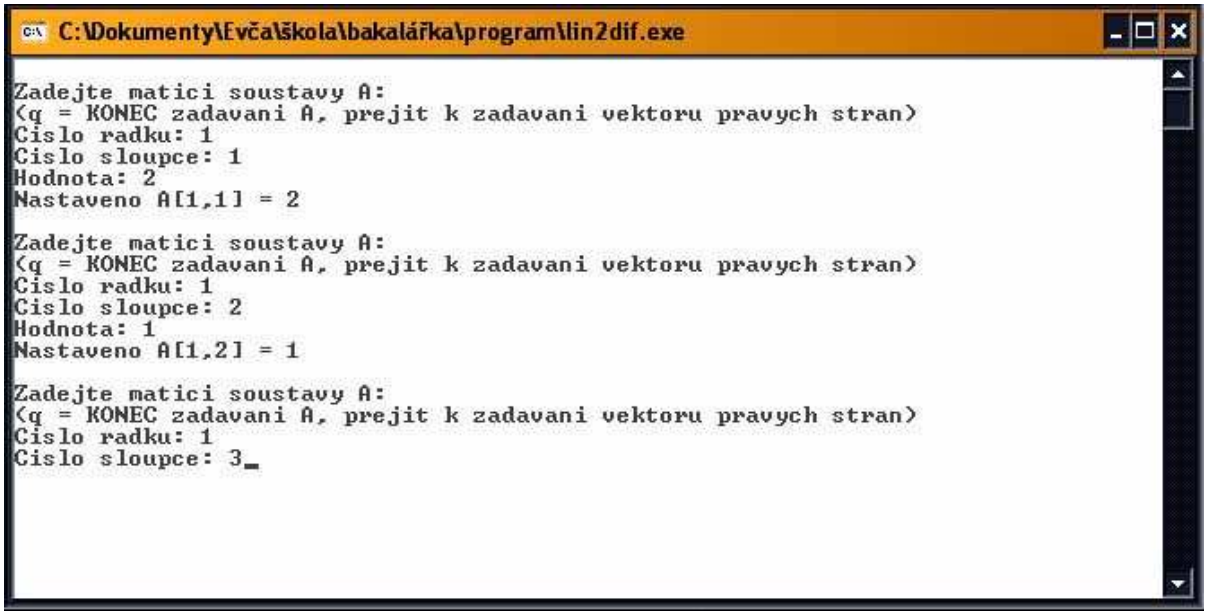
6.3 Ukázka práce s programem

Funkčnost aplikace ukážeme na soustavě lineárních rovnic:

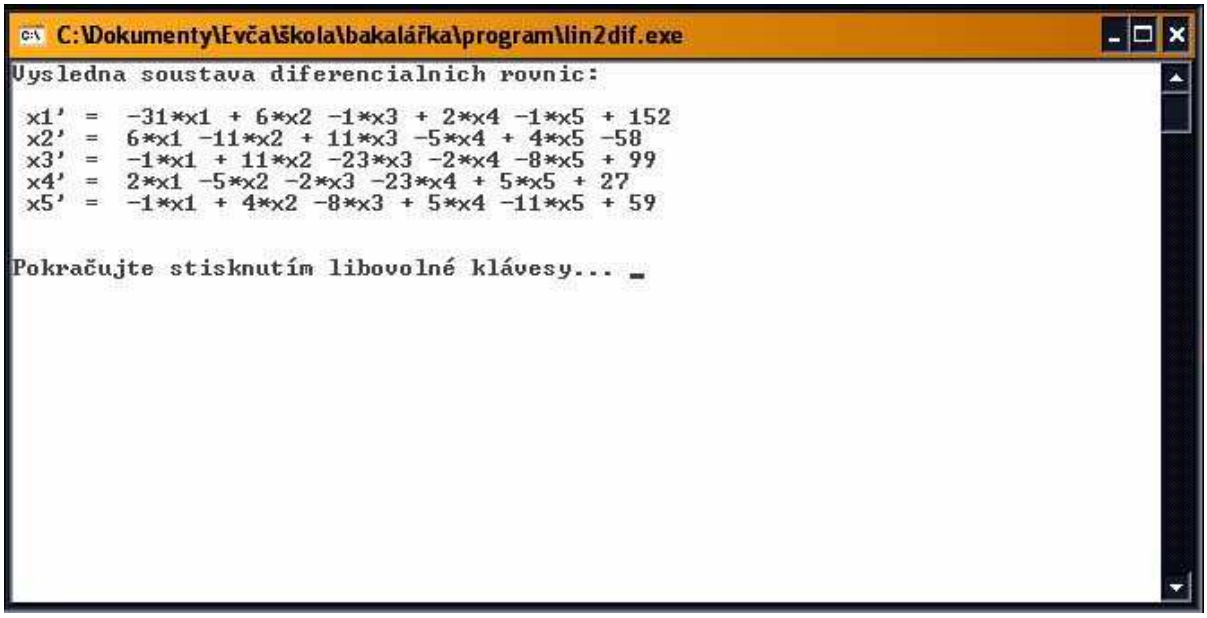
$$2x_1 + x_2 - x_3 + 4x_4 - x_5 = 12$$
$$x_1 - 2x_2 + x_3 - x_4 + 2x_5 = 12$$
$$3x_1 + x_2 - 2x_3 - 2x_4 + x_5 = 10$$
$$4x_1 - 2x_2 + x_3 - x_4 - x_5 = 15$$
$$x_1 - x_2 + 4x_3 + x_4 + 2x_5 = 26$$

Po spuštění programu jsme vyzvaní, abychom zadali matici soustavy. Nejprve zadáme číslo řádku, potvrdíme enterem, to samé u sloupce a hodnoty prvku. Analogicky pokračujeme, dokud nezadáme všechny koeficienty matice soustavy, pak stiskneme klávesu q a opět potvrdíme enterem.

Po stisknutí klávesy q můžeme zadávat vektor pravých stran. Zde zadáváme pouze číslo řádku a hodnotu. Po ukončení zadávání opět stiskneme klávesu q, tím spustíme výpočet a v dalším okně se nám zobrazí výsledné rovnice. K ukončení programu stiskneme libovolnou klávesu nebo použijeme křížek v pravém horním rohu. Vygenerované rovnice si můžeme zkopírovat z aplikace nebo je použít z výstupního souboru vysledek.txt, který se uloží do adresáře, ve kterém máme program uložený.



Obrázek 6.1 Zadávání jednotlivých prvků



Obrázek 6.2 Zobrazení výsledné soustavy rovnic

7 Závěr

Cílem této práce bylo seznámení se s výpočtem soustav lineárních algebraických rovnic především pomocí převodu na soustavu rovnic diferenciálních a ověření správnosti výsledků. K ulehčení převodu mezi soustavami byl vytvořen jednoduchý program lin2dif.

Závěrem můžeme říct, že výsledky získané pomocí metody převodu závisí na vlastnostech matice soustavy. Např. velikost chyby závisí na hodnotě determinantu. Čím více se determinant blíží nule, tím jsou získané hodnoty nepřesnější. Srovnáme-li metodu převodu na diferenciální rovnice s iteračními metodami, zjistíme, že metoda převodu je rychlejší, protože jednotlivé iterace mohou být počítané paralelně. Pro soustavy, které obsahují málo rovnic je zase vhodnější použít numerické metody. Zřejmě také můžeme říci, že řešíme-li soustavu pomocí převodu, dostaneme řešení s větší pravděpodobností, než kdybychom ji řešili jinými metodami.

Soustavy diferenciálních rovnic byly řešeny v programu TKSL, ale existuje i novější verze TKSL/C, která je napsaná v jazyce C++. Do budoucna by tedy bylo možné propojit program TKSL/C s programem lin2dif, který by mohl být ještě rozšířen a zdokonalen.

Literatura

[1] Rektorys, K.: Přehled užité matematiky, SNTL, Praha, 1996.

[2] Fajmon, B., Růžičková, I.: Matematika 3, Vysoké učení technické v Brně – Fakulta elektrotechniky a komunikačních technologií, Brno.

[3] Kovár, M.: Maticový a tenzorový počet, Vysoké učení technické v Brně – Fakulta elektrotechniky a komunikačních technologií, Brno.

[4] Dostál, Z.: Lineární algebra, Vysoká škola báňská – Technická univerzita Ostrava, Ostrava, 2000.

[5] Vicher, M.: Numerická matematika, 2003

[6] WWW stránky: Matematika online. http://mathonline.fme.vutbr.cz/

[7] WWW stránky: High performance computers. http://www.fit.vutbr.cz/~kunovsky/TKSL/

Seznam příloh

Příloha 1. Manuál programu lin2dif

1. Spustťe aplikaci lin2dif.exe.
2. Zadejte prvky matice soustavy. Zadávat se smí pouze celá čísla, jsou-li prvky nulové, zadávat se nemusí. Pro potvrzení zadávaního čísla stiskněte enter. Pro kontrolu se vypíší koeficienty zadaného prvku a jeho hodnota. Po ukončení zadávání stiskněte q a potvrďte enterem.
3. Zadejte prvky vektoru pravých stran. Pro zadávání hodnot platí totéž jako v předcházejícím bodě. Po zadání všech hodnost stiskněte q a potvrďte enterem. Následuje převod.
4. Zobrazí se výsledná soustava. Aplikaci ukončete stisknutím libovolné klávesy nebo kliknutím na křížek v pravém horním rohu.
5. Výsledek je uložen do souboru vysledek.txt, který se nachází ve stejné složce jako aplikace lin2dif.exe.

Příloha 2. Zdrojový text programu lin2dif

```
/* Soubor:  lin2dif.cpp
 * Datum:   25.4.2009
 * Autor:   Eva Kucharova, xkucha17@stud.fit.vutbr.cz
 * Popis:    Program umoznujici prevod soustav linearnich algebraickych rovnic na soustavu rovnic diferencialnich
*/

#include <cstdlib>
#include <cstdio>
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>

using namespace std;
const int SIZE_SLOUPEC = 100;
const int SIZE_RADEK = 100;

int A[SIZE_SLOUPEC][SIZE_RADEK] = {0}; // staticka vynulovana matice o rozmerech 100x100
int B[1][SIZE_RADEK] = {0};      // staticka vynulovana matice o rozmerech 1x100

void transponuj_matici(int mat[][SIZE_RADEK], int sloupec)
{
    int posun = 0, tmp = 0;
    for(int i =0; i< sloupec; i++)
    {
        for(int j = posun; j< SIZE_RADEK; j++)
        {
            tmp = mat[j][i];
            mat[j][i] = mat[i][j];
            mat[i][j] = tmp;
        } posun++;
    }

}
void    nasobeni_matice(int    m[][SIZE_RADEK],    int    ma[][SIZE_RADEK],    int mb[][SIZE_RADEK], int ma_slo, int mb_slo)
{
    int m_rad = SIZE_RADEK, m_slo = mb_slo;

    for (int i = 0; i < SIZE_RADEK; i++)
```

```
        for (int j = 0; j < mb_slo; j++)
        {
            m[j][i] = 0;
            for (int k = 0; k < ma_slo; k++)
            {
                 m[j][i] += ma[k][i] * mb[j][k];
            }
        }
}

void kopiruj_matici(int mat[][SIZE_RADEK], int mat2[][SIZE_RADEK] , int sloupec)
{
      for(int i =0; i< sloupec; i++)
          for(int j =0; j< SIZE_RADEK; j++)
              mat2[i][j] = mat[i][j];
}

void neguj_matici(int mat[][SIZE_RADEK], int sloupec)
{
      for(int i =0; i< sloupec; i++)
          for(int j =0; j< SIZE_RADEK; j++)
              mat[i][j] *= -1;
}

void nacti_matici()
{
    char tmp[100];
    int cislo, radek, sloupec, hodnota;
    string retez;
    bool ulozeno = false;
                                        //  nacitani matice A
    while(retez.compare("q") != 0)
    {
        if(!ulozeno)
        system("cls");                //     smazani obrazovky
        cout << "\nZadejte matici soustavy A: \n(q = KONEC zadavani A, prejit k
zadavani vektoru pravych stran)"<<endl;

                                        //  zadavani radku
        cout <<"Cislo radku: ";
        cin >> retez;                   //  nacteni hodnoty z klavesnice
        fflush(stdin);

        istringstream myStream(retez);  //  prevedeni na int

        if (myStream >> cislo)          //  pokud se prevod povedl
```

```
{
      snprintf(tmp, 100 ,"%d",cislo);

      if(cislo<0 || cislo >=SIZE_RADEK  || strlen(tmp)!= retez.length())
      {
          ulozeno = false;
          continue;
      }else
      {
          radek = cislo;
          radek--;
      }
}else
{
      ulozeno = false;
      continue;
}
                                // zadavani sloupce
cout <<"Cislo sloupce: ";
cin >> retez;                   // nacteni hodnoty z klavesnice
fflush(stdin);

istringstream myStream2(retez);    // prevedeni na int

if (myStream2 >> cislo)  // pokud se prevod povedl
{
       snprintf(tmp, 100 ,"%d",cislo);

      if(cislo<0 || cislo >=SIZE_SLOUPEC  || strlen(tmp)!= retez.length())
      {
          ulozeno = false;
          continue;
      }else
      {
        sloupec = cislo;
        sloupec--;
      }
}else
{
    ulozeno = false;
    continue;
}

                                // zadavani hodnoty
cout <<"Hodnota: ";
cin >> retez;                   // nacteni hodnoty z klavesnice
```

```
        fflush(stdin);

        istringstream myStream3(retez); // prevedeni na int

        if (myStream3 >> cislo)         // pokud se prevod povedl
        snprintf(tmp, 100 ,"%d",cislo);
        if(strlen(tmp)== retez.length())
        {
            hodnota = cislo;
            A[sloupec][radek] = hodnota;
            cout<<"Nastaveno A["<<radek+1<<","<<sloupec+1<<"] = "<<hodnota<<endl;
            ulozeno = true;

        }else
        {
            ulozeno = false;
            continue;
        }
    }


    ulozeno = false;
    retez = "";

                                    // zadavani matice B
    while(retez.compare("q") != 0)
    {
        if(!ulozeno)
          system("cls");               // smazani obrazovky
        cout << "\nZadejte vektor pravych stran B: \n(q = KONEC zadavani B, prejit
k prevodu soustavy rovic)"<<endl;

                                    // zadavani radku
        cout <<"Cislo radku: ";
        cin >> retez;                   // nacteni hodnoty z klavesnice
        fflush(stdin);

        istringstream myStream(retez);  // prevedeni na int

        if (myStream >> cislo)          // pokud se prevod povedl
        {
              snprintf(tmp, 100 ,"%d",cislo);

              if(cislo<0 || cislo >=SIZE_RADEK  || strlen(tmp)!= retez.length())
              {
                  ulozeno = false;
```

```
                continue;
            }else
            {
                radek = cislo;
                radek--;
            }
        }else
        {
            ulozeno = false;
            continue;
        }
                                    // zadavani hodnoty
        cout <<"Hodnota: ";
        cin >> retez;                   // nacteni hodnoty z klavesnice
        fflush(stdin);

        istringstream myStream3(retez);   // prevedeni na int

        if (myStream3 >> cislo)           // pokud se prevod povedl
        snprintf(tmp, 100 ,"%d",cislo);
        if(strlen(tmp)== retez.length())
        {
            hodnota = cislo;
            B[0][radek] = hodnota;
            cout<<"Nastaveno B["<<radek+1<<", 1] = "<<hodnota<<endl;
            ulozeno = true;

        }else
        {
            ulozeno = false;
            continue;
        }

    }
}

void vypis_matici(int mat[][SIZE_RADEK],int sloupec, string label)
{
  cout<< label<<":"<<endl;
  for(int i =0; i< sloupec; i++)
      for(int j =0; j< SIZE_RADEK; j++)
      {
          if(mat[i][j] != 0)
            cout<<"["<<i<<"]["<<j<<"] = "<<  mat[i][j]<<endl;
      }
}
```

```
void setRadky(string radek[SIZE_RADEK], int C[][SIZE_RADEK], int D[][SIZE_RADEK])
{
    bool nasel = false, prvniSloupec = true;

    char buff[100];
    int index = 0;
    for(int j =0; j< SIZE_RADEK; j++)
    {
            nasel = false;       // nasel nenulovou hodnotu v matici A
            prvniSloupec = true;    // nevypisovani '+' u prvniho sloupce

            for(int i =0; i< SIZE_SLOUPEC; i++)
            {
                if(C[i][j] > 0 && prvniSloupec) // prvni sloupec je kladny
                {
                    sprintf(buff, " %d*x%d", C[i][j],i+1);
                    radek[index] += buff;

                    nasel = true;
                    prvniSloupec = false;

                }else if(C[i][j] > 0 && !prvniSloupec)    // dalsi kladny sloupec
                {

                    sprintf(buff, " + %d*x%d", C[i][j],i+1);
                    radek[index] += buff;

                    nasel = true;
                    prvniSloupec = false;

                }else if (C[i][j] < 0)                  // zaporny sloupec
                {
                    sprintf(buff, " %d*x%d", C[i][j],i+1);
                    radek[index] += buff;

                    nasel = true;
                    prvniSloupec = false;
                }
            }

            if(nasel)        // pokud bylo nalezeno cislo, vypisi matici B
            {
                if(D[0][j] >= 0)   // pokud je hodnota v B kladna nebo rovna nule
                {
                    sprintf(buff, " + %d ", D[0][j]);
```

```
                    radek[index] += buff;

                    sprintf(buff, " x%d' = ", j+1);
                    radek[index] = buff+radek[index];

                    index++;

                }else if (D[0][j] < 0)          // pokud je hodnota v B zaporna
                {
                    sprintf(buff, " %d ", D[0][j]);
                    radek[index] += buff;

                    sprintf(buff, " x%d' = ", j+1);
                    radek[index] = buff+radek[index];

                    index++;
                }
            }
    }
}

void vypis_vysledek(int C[][SIZE_RADEK], int D[][SIZE_RADEK])
{
    string radek[SIZE_RADEK] = {""};
    setRadky(radek, C, D);

    //vypis
    for(int i =0;i<SIZE_RADEK; i++)
    {
             if(radek[i].compare("") != 0)
                   cout<<radek[i]<<endl;
    }
}

void   vypis_vysledek_do(const   char   *filename,   int   C[][SIZE_RADEK],   int
D[][SIZE_RADEK])
{
    string radek[SIZE_RADEK] = {""};

    setRadky(radek, C, D);

    ofstream myfile;  // vypis do souboru
       myfile.open (filename, ios::out);  // pro pridani zmenit na ios::app

    for(int i =0;i<SIZE_RADEK; i++)
    {
```

```
        if(radek[i].compare("") != 0)
                    myfile<<radek[i]<<endl;
    }
    myfile.close();
}

int main(int argc, char *argv[])
{
    nacti_matici();
    system("cls");        // smazani obrazovky

    int T[SIZE_SLOUPEC][SIZE_RADEK] = {0};
    kopiruj_matici(A, T , SIZE_SLOUPEC);  // zkopirovani matice A do matice T

    transponuj_matici(T, SIZE_SLOUPEC);  // transponovani matice T

    int C[SIZE_SLOUPEC][SIZE_RADEK] = {0};
    nasobeni_matice(C,   T,   A,   SIZE_SLOUPEC,   SIZE_SLOUPEC);      //   nasobeni
transponovane matice T s matici A

    neguj_matici(C, SIZE_SLOUPEC);  // vysledek promennych x1,x2, ... , x100

    int D[1][SIZE_RADEK] = {0};

    nasobeni_matice(D, T, B, SIZE_SLOUPEC, 1);  // vynasobeni transponovane matice
T s matici B

    cout<<"Vysledna soustava diferencialnich rovnic:\n"<<endl;
    vypis_vysledek(C, D);    // vypsani vysledku na obrazovku
    cout<<"\n"<<endl;
    vypis_vysledek_do("vysledek.txt", C, D);     // ulozeni vysledku do souboru
    system("PAUSE");
    return EXIT_SUCCESS;
}
```